ScienceDirect

Disponible en www.sciencedirect.com

Revista Iberoamericana de Automática e Informática industrial 13 (2016) 363–369

www.elsevier.es/RIAI

Integración automática de dispositivos en el Hogar Digital a través de la generación de adaptadores dirigida por modelos

María Rodríguez [a, *], Eduardo Zalama [b], Ignacio González [c]

[a] *Departamento de Ingeniería de Sistemas y Automática, Universidad de Valladolid, Paseo del Cauce S/N 47011, Valladolid, España*
[b] *Instituto de las Tecnologías Avanzadas de la Producción, Universidad de Valladolid, Paseo del Cauce S/N 47011, Valladolid, España*
[c] *Departamento de Informática, Universidad de Oviedo, C/Gonzalo Gutiérrez Quirós S/N, Mieres, Asturias, España*

Resumen

En el contexto del Hogar Digital, donde múltiples sistemas de muy diversa índole tienen que trabajar conjuntamente, se han hecho muchos esfuerzos encaminados a conseguir un estándar de comunicaciones que permita la interoperabilidad entre los mismos. El protocolo Digital Home Compliant (DHC) persigue ese objetivo creando una red que ofrece servicios comunes a los distintos dispositivos como son los de localización, eficiencia energética o seguridad. Para formar parte de la red DHC y poder interactuar con ella, es necesario el desarrollo de un adaptador software para cada dispositivo que actúa como intermediario con la red. Dado que dichos adaptadores tienen una proporción de código común bastante alta, y que el código específico que permite controlar el dispositivo suele venir dado por el fabricante, se plantea en este trabajo automatizar el proceso de generación de adaptadores, siguiendo el paradigma de la Arquitectura Dirigida por Modelos (ADM). Una vez generado el código partir de una plantilla y del modelo SysML del sistema, el adaptador se puede desplegar automáticamente, y como consecuencia, mejorar la interoperabilidad del sistema. Para ilustrar las ventajas de la propuesta, se plantea el caso de estudio particular de la generación del adaptador DHC para el robot de servicio Roomba.

Palabras Clave:
Código, generación, interacción, robots, modelos de sistemas

1. Introducción

Un Hogar Digital puede definirse como un entorno inteligente que tiene la capacidad de adaptarse a sus ocupantes. Para poder hacerlo, los elementos que lo componen – electrodomésticos inteligentes, robots de servicio, así como otros dispositivos tradicionales – deben poder comunicarse entre ellos para realizar las tareas de forma colaborativa. Otra parte importante de este tipo de entornos que cada vez toma más fuerza es la gestión eficiente de la energía. En la Agenda estratégica de la Plataforma española de Redes Eléctricas (Futured | Plataforma española de redes eléctricas, 2012) se resalta que el principal obstáculo para medir, gestionar y anticipar la demanda de energía es la ausencia de un protocolo abierto y estándar que asegure la interoperabilidad entre dispositivos heterogéneos de distintos fabricantes, que además pueden tener propósitos diferentes (entretenimiento, vigilancia, gestión de la energía, labores del hogar o asistencia personal) (Perumal et al., 2008).

El protocolo de comunicaciones Digital Home Compliant (DHC) (González Alonso et al., 2012) – actualmente en su versión 2 – surge con el objetivo de cubrir esta necesidad, ofreciendo una forma estandarizada para que los citados dispositivos puedan ofrecer sus servicios en ese contexto de hogar inteligente.

Los dispositivos que cumplan esta característica tendrían una etiqueta distintiva (Figura 1) certificando que cumplen con el estándar, de forma que el usuario los pueda identificar en el mercado, permitiendo a los fabricantes de estos dispositivos distinguirse entre la competencia.

Para formar parte de la red DHC es necesario implementar un adaptador software, específico para cada dispositivo, que actúa como intermediario entre dicho dispositivo y la red DHC, y que le va a permitir trabajar conjuntamente con el resto de dispositivos del hogar.

Figura 1: DHCompliant label

* Autor en correspondencia.
Correos electrónicos: maria.rodriguez.fernandez@gmail.com (Primer A. Autor)

http://dx.doi.org/10.1016/j.riai.2016.03.007

La idea de partida es el hecho de que todos los adaptadores tienen una proporción muy alta de código en común (por ejemplo, el módulo destinado a la gestión de la energía (DHC Energy) está compuesto por 1800 líneas de código, de las cuales sólo 250 son específicas del dispositivo), por lo que se propone una herramienta para generar automáticamente dicho código común y facilitar la integración de la parte de código específica, estandarizando el proceso. La propuesta sigue las bases de la Arquitectura Dirigida por Modelos (ADM), respaldada por el grupo *Object Management Group* (OMG) (Selic, 2003) para la construcción de software heterogéneo.

Siguiendo esta filosofía, la elección del método de modelado será crucial, especialmente en un sistema con componentes de tan diversa índole como es el tratado. Además, a la hora de diseñar la arquitectura del mismo, el uso de patrones de diseño que faciliten su escalabilidad repercutirá notablemente en la reducción de tiempo en el desarrollo de futuros adaptadores.

Será igualmente importante el estudio de la técnica, ya que al tener que integrarse el adaptador generado en un protocolo existente, partimos con una serie de restricciones (lenguaje, arquitectura, etc.) a las que la herramienta de generación de código usada tendría que poder ajustarse.

Este documento está organizado de la siguiente manera: Primeramente, en la sección Estado del Arte, se verán las tecnologías más relevantes y se hará referencia a otros estudios relacionados. A continuación, se pasa a describir la solución propuesta, ilustrada con el desarrollo de un experimento. Finalmente, el documento se cierra con las conclusiones y las líneas futuras de investigación.

2. Estado del Arte

Con el objetivo de cubrir todos los aspectos relacionados con esta investigación, primeramente se mostrará una visión general sobre las tecnologías de interoperabilidad existentes aplicables al hogar, tanto específicas, como arquitecturas software, donde se enmarcaría el protocolo DHC. Focalizando un paso más hacia el objetivo de la investigación, se hace un repaso por otros trabajos donde se ha aplicado el enfoque de ADM y que puedan tener aspectos en común con el objetivo de este trabajo. Por último se realiza un estudio tecnológico de las herramientas de generación de código, a través de una comparativa y la correspondiente discusión, de cara a ilustrar la selección tecnológica de la solución.

2.1. Tecnologías para la interoperabilidad

Desde los comienzos de la domótica diversas agrupaciones de empresas han tratado de establecer estándares de fabricación que permitiesen la interoperabilidad de los distintos dispositivos de un hogar digital, de los cuales destacan KNX (KNX Association, n.d.), que unifica protocolos domóticos en Europa, y Lonworks (Loy et al., 2012), sistema propietario más indicado para la automatización a gran escala que para un hogar. Ambos son soluciones específicas, caras para aplicarse en un hogar, con un carácter muy cerrado y muchas restricciones.

En la actualizada, la tendencia del sector va encaminada al desarrollo de arquitecturas independientes del medio físico empleado. Así surgen diversos protocolos de comunicación estándar que permiten el intercambio de información entre cualquier dispositivo conectado a la red. Los más extendidos son uPnP (Universal Plug and Play) (Allard et al., 2003), Jini (Furmento et al., 2002), Echonet (Energy Conservation and Homecare Network) (Echonet Consortium, 2013) o HAVi (Home Audio/Video interoperability) (Teirikangas, 2001). Estas tecnologías tienen en común que proporcionan la opción de intercomunicar dispositivos y servicios heterogéneos. Existen varios estudios donde se comparan estos protocolos, (Talal and Rachid, 2013) (Lee and Helal, 2002) haciendo especial hincapié en valorar su robustez, el mecanismo de selección del servicio, la adaptabilidad y por supuesto, la interoperabilidad. La tecnología que está siendo cada vez más adoptada por este tipo de entornos es la Arquitectura Orientada a Servicios (SOA, siglas del inglés Service Oriented Architecture), ya que ofrece una forma bien definida de exponer servicios (normalmente Servicios Web) facilitando la interacción entre diferentes sistemas. Este enfoque proporciona flexibilidad, escalabilidad y reusabilidad, así como ahorros de tiempo y dinero cuando se desarrolla una aplicación (Papazoglou and Van Den Heuvel, 2007).

Dentro de SOA, tienen cabida diversas tecnologías. En concreto, la tecnología Devices Profile for Web Services (DPWS) (Cândido et al., 2010) permite definir un perfil para habilitar mensajes, descubrimiento, descripción y eventos de forma segura.

DPWS se basa en las siguientes tecnologías XML, definidas por el World Wide Web Consortium (W3C) (Consortium, 2003):

- Simple Object Access Protocol (SOAP) para formatear los mensajes basados en XML.
- Web Services Description Language (WSDL) para describir la funcionalidad de los servicios web.
- XML-Schema para describir los mensajes intercambiados por los servicios web.
- Hypertext Transport Protocol (HTTP) como protocolo de transporte.

La idoneidad de DPWS para desarrollar una arquitectura flexible en el hogar digital ya ha sido probada en otros estudios (Parra et al., 2009). Además, la ubicuidad de HTML y XML permite un nivel alto de interoperabilidad. El método presentado en este artículo cumple con los estándares mencionados.

2.2. Trabajos relacionados

El principal objetivo de ADM es lograr generar sistemas a partir de modelos definidos a alto nivel.

Para realizar el modelado, la OMG recomienda el uso del Lenguaje de Modelado Unificado (UML) (Kleppe et al., 2003). Debido a la particularidad del sistema propuesto en este estudio, que integra tanto componentes software como hardware, se ha considerado más adecuado el uso de SysML, que siguiendo las pautas de UML – lenguaje en el que está basado - está más indicado para el modelado de sistemas (Hause and Thom, 2008) (Chang et al., 2011).

El paradigma ADM ha sido aplicado a la generación de código en otros campos similares, entre los que cabe destacar el desarrollo de software robótico (Rahman et al., 2011). También se ha aplicado en entornos inteligentes pero sobre otros protocolos como el ya citado uPnP (Sulistyo and Prinz, 2009). Su idoneidad en la generación de código también ha sido estudiada y probada exitosamente en varias investigaciones, en campos tan exigentes como los sistemas en tiempo real empotrados (Vidal et al., 2009) (Raslan and Sameh, 2007).

En cuanto al sector de los edificios energéticamente eficientes, son múltiples los intentos de resolver la falta de interoperabilidad entre las diferentes tecnologías que toman parte en este escenario. Este tipo de proyectos - como AMIGO o Hydra - proponen desarrollar una capa middleware software totalmente transparente al usuario (Warriach, 2013). Para el desarrollo de dicha capa middleware, hay trabajos que han utilizado generación automática de código. En concreto, el proyecto EnTiMid (Nain et al., 2008) se basa en el supuesto de que existe una capa de middleware única y universal que proporcionaría soporte para varias tecnologías basadas en servicios entre las que se encuentra DPWS. Por su parte Milanovik realiza un modelado semántico con ciertas restricciones, como que el dominio del modelo y de la ontología sea idéntico (Ribarić et al., 2008). Kaed va un paso más allá, añadiendo ADM al modelado semántico del sistema (Kaed et al., 2010).

Por último, la conversión entre UML y WSDL ha sido estudiada y probada para la generación de Servicios Web en varios trabajos (Emig et al., 2007) (Jongmans et al., 2012).

2.3. Herramientas de generación de código

Otro aspecto a considerar es el tipo y la calidad del resultado generado. La generación de código Java a partir de modelos no es ninguna novedad y es una opción que integran la mayoría de entornos de desarrollo (Harrison et al., 2000).

En cuanto al estado de la tecnología, se ha realizado un estudio de herramientas de generación de código, para valorar la más adecuada a la hora de realizar el adaptador.

Concretamente se han analizado las herramientas listadas en la Tabla 1.

Tabla 1: Comparación de herramientas de generación de código

Nombre	Gratuita	Entrada	Salida	Eclipse
Acceleo	SI	EMF based model and templates	Any	SI
Actifsource	SI	UML Model and templates	Any	SI
Any Code Astah plugin	NO	UML Model and templates	Any	NO
Code-g	SI	Pattern based code	Any	NO
CodeSmith Generator	NO	Template	Any	NO
GenerateXY	NO	Model, XML, CSV, Java, etc.	Any	NO
Jinja	SI	Template	Python	NO
OMS.Ice	SI		Any	SI
ZumCoder	NO	Template	C#, PHP, ASP, Java, SQL, MySQL	NO

La primera restricción a tener en cuenta es que debe soportar UML y preferentemente SysML como entrada. Como se ve en la tabla, la mayoría usan versiones anteriores a UML2.x, y no ofrecen soporte para diagramas SysML.

En cuanto a la salida debería dar soporte para lenguaje Java y WSDL por compatibilidad con el protocolo DHC. Quedarían descartadas entonces Code-g, Jinja y ZumCoder.

También se ha tenido en cuenta el tipo de licencia del software, valorando positivamente las herramientas libres.

Por último, la integración con la herramienta de desarrollo Eclipse es deseable aunque no imprescindible, debido a que permitiría integrar el modelado, la realización de plantillas y las funcionalidades de generación de código en un único entorno.

Teniendo en cuenta lo anterior, de las herramientas estudiadas, se selecciona como más adecuada Acceleo, que será tratada con más detalle en la siguiente sección.

3. Solución propuesta

Esta sección se divide en tres partes. Primeramente, es necesario ofrecer unos conceptos generales de la arquitectura del protocolo DHC, para poder comprender el modelado del sistema que se hará a continuación. En la segunda parte se verá la forma en la que se extrae la información necesaria de dicho modelo por medio de una plantilla para generar el código correspondiente. Finalmente se ilustrará la solución propuesta con el caso particular del robot de servicio Roomba.

3.1. Modelado de la arquitectura DHC

La arquitectura del protocolo DHC (Otero et al., 2015) ha tenido una gran influencia en el modo en el que se diseñó e implementó el sistema de generación automática de adaptadores propuesto.

El diagrama SysML mostrado en la Figura 2 representa la arquitectura DHC. En este tipo de diagramas llamados “de bloques”, el elemento “block” representa la unidad básica de estructura, que puede ser tanto hardware, como software, información, etc.

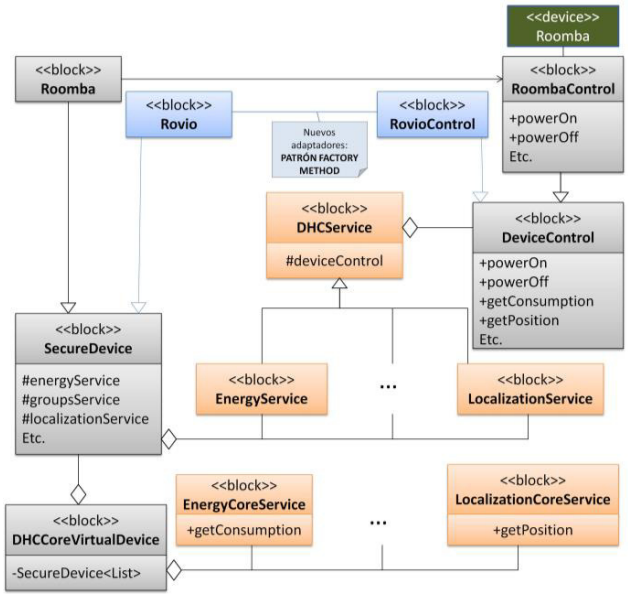

Figura 2: Arquitectura de DHC2.1

Además gracias al modelado con SysML es posible incorporar otra información como atributos y operaciones (en el diagrama sólo se muestra un subconjunto representativo de los mismos).

Como se ve, el sistema se compone de cinco sub-módulos:

- *Servicio de seguridad (DHC-Security & Privacy service):* Se encarga de la prevención del uso fraudulento de los dispositivos, así como del acceso a datos por agentes no autorizados.
- *Servicio de grupos (DHC-Groups service):* Permite la coordinación de tareas colaborativas realizadas por varios dispositivos pertenecientes a la red.
- *Servicio de Localización (DHC-Localization service):* Proporciona información de localización sobre los dispositivos de la casa, facilitando entre otras cosas la navegación de aquellos que pueden moverse.
- *Servicio de Inteligencia (DHC-Intelligence service)*: La inteligencia es integrada en el sistema por medio de un componente de gestión de reglas que controlan el funcionamiento del sistema, así como la aplicación de técnicas de aprendizaje automático para el reconocimiento de patrones de comportamiento, realizando predicciones que permitan anticiparse a ciertas acciones.
- *Servicio de energía (DHC-Energy service):* Incorpora los conceptos de energía y Smart Grid (Chen et al., 2009) al protocolo DHC, ofreciendo al usuario información sobre su consumo, con el objetivo de ayudarle a mejorar la eficiencia energética de su hogar y consecuentemente ahorrar en su factura eléctrica (Fischer, 2008).

Técnicamente, las operaciones ofrecidas por los adaptadores DHC son expuestas a través de Servicios Web gracias a la tecnología DPWS. En concreto, el conjunto de herramientas WS4D (del inglés *Web Service for Device*) ofrece el soporte necesario para la implementación de los mismos en Java.

Por una parte, por cada dispositivo de la red DHC, tiene que crearse un objeto de DPWS (denominado "Secure Device" – ver Figura 2 –) que ofrecerá los servicios DHC (energía, grupos, inteligencia, localización y seguridad) así como los servicios específicos del dispositivo. La funcionalidad será llevada a cabo en última instancia por un controlador del dispositivo (específico para el mismo, por ejemplo RooombaControl o RovioControl serían los controladores para la Roomba y el Rovio, respectivamente). Esta parte más cercana al hardware es normalmente proporcionada por el fabricante.

Por otra parte, hay parte de la funcionalidad que no puede ser manejada por el controlador, por ejemplo, un dispositivo sin mecanismos de localización como GPS no es capaz de ofrecer su posición o saber dónde está posicionado otro dispositivo que pueda tener que trabajar colaborativamente con él. En ese caso, el protocolo DHC delega en un núcleo distribuido que ofrece un conjunto de operaciones globales, que son comunes y accesibles por todos los componentes de la red DHC. Estas operaciones son expuestas como servicios de un dispositivo virtual (*Virtual Device*). Cada servicio global del *VirtualDevice* se relacionada con uno de los ya mencionados servicios asociados al dispositivo Secure Device.

El diseño se ha realizado siguiendo el patrón de diseño *Factory*, que permite reutilizar gran parte del código en la incorporación de nuevos adaptadores.

3.2. Diseño de las plantillas y generación de código

El modelo SysML descrito en el apartado anterior es común para todos los procesos de generación de adaptadores, mientras que la plantilla tiene que ser definida de forma específica para cada adaptador. Como se ve en la Figura 3, una vez identificado el dispositivo, se seleccionaría la plantilla adecuada para el mismo, y junto con el modelo del sistema, compondrían la entrada del generador. Como se vio en la sección de Estado del Arte, la herramienta seleccionada para realizar la generación de código fue Acceleo (Musset et al., 2006). Se trata de una tecnología de código abierto y multiplataforma que, cumpliendo con el estándar de la OMG Model-to-Text (M2T), integra en un mismo entorno las tres funcionalidades necesarias: modelado, plantilla y generación de código.

Internamente, la plantilla obtiene la información necesaria del modelo SysML, crea el código fuente y genera los ficheros java y WSDL, donde se describen los servicios. Como ventaja adicional, Acceleo permite generar un plug-in personalizado que facilita las generaciones de adaptadores posteriores.

Como se ve, con el método propuesto, el proceso de integrar un dispositivo en la red DHC es casi transparente para el usuario, con lo que la interoperabilidad del sistema completo se ve claramente mejorada.

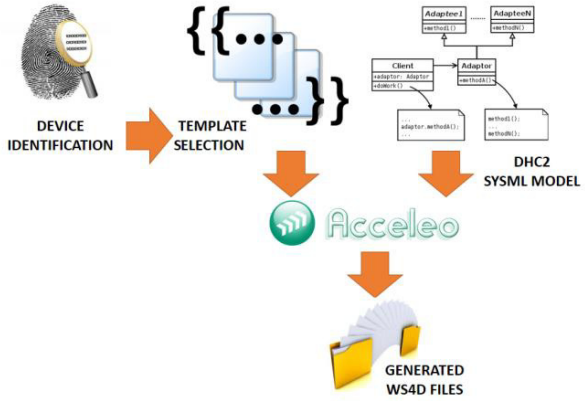

Figura 3: Proceso de generación

3.3. Caso de estudio: Adaptador de la Roomba

Para ilustrar el mecanismo de generación de adaptadores DHC propuesto, en esta sección se presenta el caso particular del robot de servicios Roomba (Tribelhorn and Dodds, 2007) (Figura 4).

Para poder controlar el robot, es necesario establecer una comunicación. En este caso, Roomba tiene un puerto mini-din donde puede conectarse un dispositivo Bluetooth o Wi-Fi. La librería java que recoge la implementación de las principales operaciones está encapsulada dentro del elemento del diagrama (Figura 2) denominado *RoombaControl*.

Figura 4: Robot de servicio Roomba

El proceso llevado a cabo a lo largo de la ejecución del experimento es mostrado gráficamente en la Figura 5 a través de un diagrama de actividad SysML. Como se ve, en cada iteración del experimento se genera el código para un dispositivo, en este caso la Roomba. Una vez ejecutado el generador de Acceleo, se obtiene el código fuente del adaptador. Si éste es válido, será desplegado automáticamente, comprobando que los Servicios Web efectivamente han sido expuestos y probando que su funcionalidad es la esperada mediante un cliente que consumirá dichos servicios.

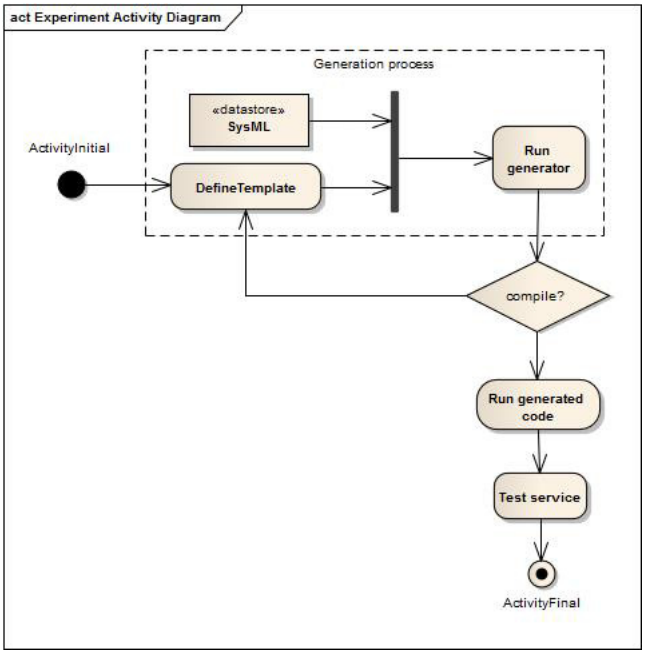

Figura 5: Diagrama de actividad SysML del proceso llevado a cabo en el experimento

La plantilla principal decide qué plantilla ha de ser aplicada para cada elemento del modelo, a través del código mostrado en la Figura 6 donde las reglas de transformación M2T están recogidas dentro de las etiquetas “template”.

```
[template public main(pack : Package)]
[comment @main/]
[if pack.name.contains('implementation')]
[for (block: Block | pack.allOwnedElements())]
     [block.generateService()  /]
     [block.generateWSDL() /]
[/for]
[/if]

[if pack.name.contains('device')]
[for (block: Block | pack.allOwnedElements())]
     [block.generateDevice()  /]
[/for]
[/if]
[/template]
```

Figura 6: Código de la plantilla principal

Además de la plantilla principal, para cada dispositivo, se han de definir tres plantillas:

- La primera genera el fichero WSDL con la definición del servicio.
- La segunda crea la implementación del servicio.
- La tercera crea el dispositivo DPWS, asociado al controlador, y por tanto al dispositivo real, el robot Roomba.

En la Figura 7, se muestra una parte representativa de la primera plantilla.

```
[template public generateWSDL (aBlock : Block)]

[let device : String = (aBlock.name.substring(0,
            aBlock.name.size()-12))]
[file ('services/'.concat(device.concat('Service.wsdl')),
       false,'UTF-8')]

<?xml version="1.0" encoding="UTF-8"?>

<wsdl:definitions targetNamespace="http://dhc2"
       ...>

  <wsdl:types>
    <xsd:schema>
      <xsd:import namespace="[aDefinition.types.namespace /]"
       schemaLocation="[aDefinition.types.schemaLocation /]"/>
    </xsd:schema>
  </wsdl:types>

  [for(m: Message | aDefinition.messages.messages)]
     <wsdl:message name ="[m.name /]">
        <wsdl:part name ="[m.part.name /]"
         element="elem:[aDefinition.linkElement(m.part) /]"/>
     </wsdl:message>

        ...

   [/for]

</wsdl:definitions>
[/file]
[/let]
[/template]
```

Figura 7: Fragmento de código de plantilla en la generación del fichero wsdl

El correspondiente código WSDL generado es mostrado en la Figura 8. Para generar este código, la plantilla se aplica sobre el modelo SysML extrayendo así los sub-módulos involucrados.

```
<?xml version="1.0" encoding="UTF-8"?>

<wsdl:definitions targetNamespace="http://dhc2"
      ...>

      <wsdl:types>
          <xsd:schema>
              <xs:element name="idDevice"
                          type="xs:string”/>
          </xsd:schema>
      </wsdl:types>

       <wsdl:message name = "CleanMessage">
            <wsdl:part element="tns:idDevice"
                  name="parameters"/>
      </wsdl:message>

       <wsdl:message name = "DockMessage">
            <wsdl:part element="tns:idDevice"
                  name="parameters"/>
      </wsdl:message>
      ...

      [/for]

</wsdl:definitions>
```

Figura 8: Fragmento del código WSDL generado

Una vez aplicadas las tres plantillas, y si todo el proceso se ha ejecutado correctamente, los servicios estarán listos para ser consumidos, y la Roomba pasa a formar parte de la red, como se ve en la Figura 9, donde se muestra el interfaz de DHC desde un televisor inteligente.

Figura 9: Interfaz de DHC2.0, con el dispositivo Roomba formando parte de la red.

La inclusión de nuevos dispositivos en la red es facilitada gracias al uso del patrón de diseño *Factory Method*, así, como se puede ver en la Figura 2, para añadir por ejemplo el robot de vigilancia *Rovio* al hogar, sólo será necesario definir el dispositivo como *SecureDevice*, y añadir el controlador, que viene dado por el propio fabricante.

3.4. Aplicación a otros ámbitos

Gracias al enfoque seguido y la selección de tecnologías realizada, la solución propuesta puede generalizarse para ser aplicada a otros ámbitos.

- El uso de SysML como lenguaje de modelado facilita la representación de cualquier tipo de sistema.
- El uso de servicios web hace que el diseño de las plantillas se pueda extrapolar a otras soluciones.
- Al ser el código generado Java, la solución será multiplataforma.

4. Conclusión y líneas futuras

El objetivo de este trabajo es mejorar la interoperabilidad entre los dispositivos perteneciente a un entorno inteligente a través del desarrollo de una herramienta de generación automática de código que, a partir de una plantilla y un modelo SysML del sistema, crea un adaptador software basado en Servicios Web, que permite al dispositivo pasar a formar parte de la red DHC.

A partir del estudio previo donde se analizaron diferentes tecnologías, la herramienta escogida fue Acceleo, ya que cumpliendo con las recomendaciones de OMG MDA, permite definir una plantilla para acceder al diagrama de arquitectura fácilmente, y obtener la información necesaria del mismo. Esta generación automática de parte del código trae como consecuencia inmediata la mejora del proceso de integración, así como la mejora del nivel de calidad y de estandarización.

La solución propuesta ha sido probada exitosamente mediante un experimento sobre el robot de servicio Roomba. Se ha logrado generar el adaptador, y el robot ha sido capaz de comunicarse con el resto de dispositivos de la red.

Respecto a las líneas futuras, sería interesante generalizar el método para otros protocolos de comunicaciones, logrando así la estandarización completa. Además, se está desarrollando un mecanismo de identificación de dispositivos para permitir reconocerlos y desplegar automáticamente el adaptador correspondiente cuando un dispositivo se integra en la red DHC.

English Summary

Improving the interoperability in the Digital Home through the automatic generation of software adapters

Abstract

In the context of the Digital Home, where multiple heterogeneous systems live together, many efforts have been made to achieve the standardization that will guarantee the interoperability among them. The Digital Home Compliant (DHC) open communication protocol arises to meet that requirement by means of a software adapter for each device that acts as intermediary with the DHC network. Based on the fact that all the adapters have a high proportion of source code in common, this paper aims to go a step further in the use of the protocol automating the adapters' process generation, following the Model Driven Architecture approach. The Web Services java code is created from a template and a SysML model of the system. The case of study of a specific adapter development – Roomba service robot – is explained in this communication to illustrate the advantages of this proposal.

Keywords:

Code, generation, interaction, robots, system models

Agradecimientos

Este trabajo ha sido posible gracias al soporte financiero ofrecido por el Departamento de Ciencia y Tecnología, Ministerio de Industria, Energía y Turismo y a los fondos FEDER. Nos gustaría agradecerle a la Universidad de Oviedo la correcta gestión del proyecto DHCompliant2 (TSI-020100-2011-313), así como a los miembros de dicho consorcio, Ingenium, Ingeniería y Domótica S.L. y Domotica DaVinci.

Referencias

Allard, J., Chinta, V., Gundala, S., Richard III, G.G., 2003. Jini meets UPnP: an architecture for Jini/UPnP interoperability, in: Applications and the Internet, 2003. Proceedings. 2003 Symposium on. pp. 268–275.

Cândido, G., Jammes, F., de Oliveira, J.B., Colombo, A.W., 2010. SOA at device level in the industrial domain: Assessment of OPC UA and DPWS specifications, in: Industrial Informatics (INDIN), 2010 8th IEEE International Conference on. IEEE, pp. 598–603.

Chang, C.-H., Lu, C.-W., Kao, K.-F., Chu, W.C., Yang, C.-T., Hsueh, N.-L., Hsiung, P.-A., Koong, C.-S., 2011. A SysML-Based Requirement Supporting Tool for Embedded Software, in: 2011 5th International Conference on Secure Software Integration Reliability Improvement Companion (SSIRI-C). Presented at the 2011 5th International Conference on Secure Software Integration Reliability Improvement Companion (SSIRI-C), pp. 202–206. DOI:10.1109/SSIRI-C.2011.34

Chen, S., Song, S., Li, L., Shen, J., 2009. Survey on smart grid technology. Power Syst. Technol. 33, 1–7.

Consortium, W.W.W., 2003. Web services architecture. W3C Work. Draft 8.

Echonet Consortium, 2013. Echonet Lite Specification.

Emig, C., Krutz, K., Link, S., Momm, C., Abeck, S., 2007. Model-driven development of SOA services. Univ. Karlsr. TH Karlsr.

Fischer, C., 2008. Feedback on household electricity consumption: a tool for saving energy? Energy Effic. 1, 79–104.

Furmento, N., Lee, W., Mayer, A., Newhouse, S., Darlington, J., 2002. ICENI: an open grid service architecture implemented with Jini, in: Proceedings of the 2002 ACM/IEEE Conference on Supercomputing. pp. 1–10.

Futured | Plataforma española de redes eléctricas, 2012. Agenda estratégica de investigación.

González Alonso, I., Álvarez Fres, O., Alonso Fernández, A., del Torno, P.G., Maestre, J.M., Almudena García Fuente, M., 2012. Towards a new open communication standard between homes and service robots, the DHCompliant case. Robot. Auton. Syst. 60, 889–900.

Harrison, W., Barton, C., Raghavachari, M., 2000. Mapping UML designs to Java. ACM.

Hause, M.C., Thom, F., 2008. An Integrated MDA Approach with SysML and UML, in: 13th IEEE International Conference on Engineering of Complex Computer Systems, 2008. ICECCS 2008. Presented at the 13th IEEE International Conference on Engineering of Complex Computer Systems, 2008. ICECCS 2008, pp. 249–254. DOI:10.1109/ICECCS.2008.21

Jongmans, S.-S.T., Santini, F., Sargolzaei, M., Arbab, F., Afsarmanesh, H., 2012. Automatic code generation for the orchestration of web services with Reo, in: Service-Oriented and Cloud Computing. Springer, pp. 1–16.

Kaed, C. El, Denneulin, Y., Ottogalli, F.-G., Mora, L.F.M., 2010. Combining ontology alignment with model driven engineering techniques for home devices interoperability, in: Software Technologies for Embedded and Ubiquitous Systems. Springer, pp. 71–82.

Kleppe, A.G., Warmer, J., Bast, W., Explained, M.D.A., 2003. The model driven architecture: practice and promise. Addison-Wesley Longman Publishing Co., Inc., Boston, MA.

KNX Association, n.d. KNX National WebSite [WWW Document]. URL https://www.knx.org/es/ (accessed 11.4.13).

Lee, C., Helal, S., 2002. Protocols for service discovery in dynamic and mobile networks. Int. J. Comput. Res. 11, 1–12.

Loy, D., Dietrich, D., Schweinzer, H.-J., 2012. Open control networks: LonWorks/EIA 709 technology. Springer Science & Business Media.

Musset, J., Juliot, É., Lacrampe, S., Piers, W., Brun, C., Goubet, L., Lussaud, Y., Allilaire, F., 2006. Acceleo user guide.

Nain, G., Daubert, E., Barais, O., Jézéquel, J.-M., 2008. Using mde to build a schizophrenic middleware for home/building automation. Springer.

Otero, A.P., Suárez, R., Varas, J.M.R., Suárez, M., Fuente, M.P.A.G., Fernández, R., Fernández, M.R., Alonso, I.G., 2015. INTEGRATION OF DIGITAL HOME, SMART APPLIANCES AND SERVICE ROBOTS USING DHCOMPLIANT 2.0. Int. J. Robot. Autom. 30. doi:10.2316/Journal.206.2015.4.206-4332

Papazoglou, M.P., Van Den Heuvel, W.-J., 2007. Service oriented architectures: approaches, technologies and research issues. VLDB J. 16, 389–415.

Parra, J., Hossain, M.A., Uribarren, A., Jacob, E., Saddik, A. El, 2009. Flexible smart home architecture using device profile for web services: a peer-to-peer approach. Int. J. Smart Home 3, 39–50.

Perumal, T., Ramli, A.R., Leong, C.Y., Mansor, S., Samsudin, K., 2008. Interoperability among Heterogeneous Systems in Smart Home Environment, in: IEEE International Conference on Signal Image Technology and Internet Based Systems, 2008. SITIS '08. Presented at the IEEE International Conference on Signal Image Technology and Internet Based Systems, 2008. SITIS '08, pp. 177–186. DOI:10.1109/SITIS.2008.94

Rahman, M.A.A., Yasuda, A., Mayama, K., Mizukawa, M., Takasu, T., 2011. Model-Driven Development of Intelligent Mobile Robot Using Systems Modeling Language (SysML). INTECH Open Access Publisher.

Raslan, W., Sameh, A., 2007. Mapping SysML to SystemC., in: FDL. pp. 225–230.

Ribarić, M., Gašević, D., Milanović, M., Giurca, A., Lukichev, S., Wagner, G., 2008. Model-Driven engineering of rules for web services, in: Generative and Transformational Techniques in Software Engineering II. Springer, pp. 377–395.

Selic, B., 2003. The pragmatics of model-driven development. Softw. IEEE 20, 19–25.

Sulistyo, S., Prinz, A., 2009. Model-Driven Development Approach for Providing Smart Home Services, in: Mokhtari, M., Khalil, I., Bauchet, J., Zhang, D., Nugent, C. (Eds.), Ambient Assistive Health and Wellness Management in the Heart of the City, Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 274–277.

Talal, B.K., Rachid, M., 2013. Service Discovery -- A Survey and Comparison. ArXiv13082912 Cs.

Teirikangas, J., 2001. HAVi: Home Audio Video Interoperability. Hels. Univ. Technol.

Tribelhorn, B., Dodds, Z., 2007. Evaluating the Roomba: A low-cost, ubiquitous platform for robotics research and education., in: ICRA. pp. 1393–1399.

Vidal, J., De Lamotte, F., Gogniat, G., Soulard, P., Diguet, J.-P., 2009. A co-design approach for embedded system modeling and code generation with UML and MARTE, in: Design, Automation & Test in Europe Conference & Exhibition, 2009. DATE'09. pp. 226–231.

Warriach, E.U., 2013. State of the art: embedded middleware platform for a smart home. Int J Smart Home 7, 275–294.